

 **UNIVERSIDAD POLITÉCNICA DE CARTAGENA**

Grado en Arquitectura Naval e Ingeniería de Sistemas Marinos

Curso académico 2025/2026

ASIGNATURA

INFORMÁTICA

COMPENDIO ACADÉMICO

LIBRO DE PRÁCTICAS

Recopilación de ejercicios resueltos

 SQL

 Python

 UPCT

AUTOR

DAVID CONESA PAGÁN

 Edición revisada |  2026



SOBRE ESTE LIBRO

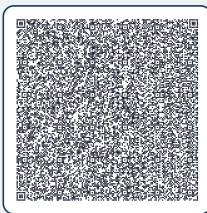
Una guía práctica para dominar la informática

Este compendio reúne la totalidad de ejercicios resueltos de la asignatura de Informática del Grado en Arquitectura Naval e Ingeniería de Sistemas Marinos. Está pensado como herramienta de estudio, repaso y consulta rápida, con código depurado, comentarios estratégicos y una presentación que pretende hacer del aprendizaje una experiencia agradable.

¿Qué encontrarás aquí?

- 🗄 Bloque SQL | consultas, agregaciones y subconsultas
- ⚙ Bloque Python | desde tipos básicos hasta funciones recursivas
- 💡 Más de 100 ejercicios resueltos paso a paso

Ⓢ AVISO LEGAL Y DERECHOS DE AUTOR



Escanea el QR
para leer el aviso legal

© 2026 David Conesa Pagán. Todos los derechos reservados. Obra original protegida por el Real Decreto Legislativo 1/1996, Ley de Propiedad Intelectual.

❗ **Obra independiente.** Apuntes personales elaborados por el autor en su condición de estudiante. **No es material oficial** ni se encuentra editado, distribuido o avalado por la Universidad Politécnica de Cartagena, su profesorado ni ningún órgano académico; las menciones a la institución y a la asignatura tienen fines exclusivamente contextuales.

🚫 **PROHIBIDA** la modificación, redistribución alterada, reventa o comercialización total o parcial de esta obra sin autorización expresa y por escrito del autor.

✓ Se permite el uso **personal, educativo y de consulta** citando autoría.

🔒 Documento firmado digitalmente **DC®** | Edición revisada | Cartagena, 2026

1 SQL	3
1.1 Práctica 0	3
1.2 Práctica 1	4
1.3 Práctica 2	8
2 Python	13
2.1 Práctica 1. Introducción al Entorno de Programación	13
2.2 Práctica 2. Tipos de Datos	14
2.3 Práctica 3. Estructuras de Control de Condicionales	18
2.4 Práctica 4. Estructuras de Control Iterativas	24
2.5 Práctica 5. Funciones I	28
2.6 Práctica 6. Funciones II	34
2.7 Práctica 7. Listas	37

PARTE 1

SQL

1.1

Práctica 0

El diagrama de entidad relación de esta práctica no se adjuntará debido a la poca probabilidad que existe que este entre, no obstante, de encontrarlo en mis archivos, se añadirá en futuras versiones del documento.

1.2 Práctica 1

C1.1 Haga la consulta SQL que muestre un listado de todas las columnas de los géneros de música (Genre).

```
SELECT *  
FROM Genre;
```

C1.2 Haga la consulta SQL que muestre un listado con los nombres de los tipos de fichero (MediaType).

```
SELECT Name  
FROM MediaType;
```

C1.3 Listado de todos los países (distintos) y sus códigos postales que aparecen en las facturas.

```
SELECT DISTINCT BillingCountry, BillingPostalCode  
FROM Invoice;
```

C1.4 Listado con el precio y nombre de canciones que duran más de 48 minutos.

```
-- 48 min = 2.880.000 ms  
SELECT Name, UnitPrice  
FROM Track  
WHERE Milliseconds > 2880000;
```

C1.5 Artistas que terminan en "ra".

```
SELECT Name  
FROM Artist  
WHERE Name LIKE '%ra';
```

C1.6 Fechas de facturas a España por importe mayor a 2.5€.

```
SELECT InvoiceDate  
FROM Invoice  
WHERE BillingCountry = 'Spain' AND Total > 2.5;
```

C1.7 Géneros de canciones con duración entre 10 y 15 minutos.

```
SELECT Name  
FROM Genre  
WHERE GenreId IN (  
    SELECT GenreId FROM Track  
    WHERE Milliseconds BETWEEN 600000 AND 900000  
);
```

C1.8 Títulos de los álbumes de U2.

```
SELECT Title  
FROM Album  
WHERE ArtistId IN (  
    SELECT ArtistId  
    FROM Artist  
    WHERE Name = 'U2'  
);
```

C1.9 Clientes con facturas en Spain, Argentina o Brazil.

```
SELECT *
FROM Customer
WHERE CustomerId IN (
    SELECT CustomerId
    FROM Invoice
    WHERE BillingCountry IN ('Spain', 'Argentina', 'Brazil')
);
```

C1.10 Canciones que NO son "AAC audio file".

```
SELECT *
FROM Track
WHERE MediaTypeId NOT IN (
    SELECT MediaTypeId FROM MediaType
    WHERE Name = 'AAC audio file'
)
ORDER BY Milliseconds;
```

C1.11 Canciones facturadas a España (Nombre y Precio).

```
SELECT Name, UnitPrice
FROM Track
WHERE TrackId IN (
    SELECT TrackId FROM InvoiceLine
    WHERE InvoiceId IN (
        SELECT InvoiceId
        FROM Invoice
        WHERE BillingCountry = 'Spain'
    )
);
```

C1.12 Artistas con canciones de Opera, Pop o World.

```
SELECT Name
FROM Artist
WHERE ArtistId IN (
    SELECT ArtistId
    FROM Album
    WHERE AlbumId IN (
        SELECT AlbumId
        FROM Track
        WHERE GenreId IN (
            SELECT GenreId
            FROM Genre
            WHERE Name IN ('Opera', 'Pop', 'World')
        )
    )
);
```

C1.13 Clientes con compra >10€ incluyendo Rock/Latin/Metal >10MB.

```
SELECT *
FROM Customer
WHERE CustomerId IN (
    SELECT CustomerId
    FROM Invoice
    WHERE Total > 10 AND InvoiceId IN (
        SELECT InvoiceId
        FROM InvoiceLine
        WHERE TrackId IN (
            SELECT TrackId
            FROM Track
            WHERE Bytes > 10485760 AND GenreId IN (
                SELECT GenreId
                FROM Genre
                WHERE Name IN ('Rock', 'Latin', 'Metal')
            )
        )
    )
);
```

C1.14 Género y canciones que empiezan por 'ra'.

```
SELECT g.Name, t.Name
FROM Genre g, Track t
WHERE g.GenreId = t.GenreId AND t.Name LIKE 'ra%';
```

C1.15 Clientes USA que han gastado más de 10€ en una compra.

```
SELECT c.FirstName, c.LastName, i.InvoiceDate, i.Total
FROM Customer c, Invoice i
WHERE c.CustomerId = i.CustomerId
AND c.Country = 'USA' AND i.Total > 10;
```

C1.16 Género y tipo de archivo de canciones que empiezan por 'ra'.

```
SELECT g.Name, m.Name
FROM Genre g, MediaType m, Track t
WHERE t.GenreId = g.GenreId AND t.MediaTypeId = m.MediaTypeId
AND t.Name LIKE 'ra%';
```

C1.17 Artista y Álbum con canciones de género 'TV Shows'.

```
SELECT ar.Name, al.Title
FROM Artist ar, Album al, Track t, Genre g
WHERE ar.ArtistId = al.ArtistId AND al.AlbumId = t.AlbumId
AND t.GenreId = g.GenreId AND g.Name = 'TV Shows';
```

C1.18 Total de ventas a clientes de España.

```
SELECT sum(Total)
FROM Invoice
WHERE BillingCountry = 'Spain';
```

C1.19 Factura de mayor precio.

```
SELECT *  
FROM Invoice  
WHERE Total = (SELECT max(Total)  
               FROM Invoice);
```

C1.20 Cuántas canciones tiene U2.

```
SELECT count(*)  
FROM Track  
WHERE AlbumId IN (  
    SELECT AlbumId  
    FROM Album  
    WHERE ArtistId IN (  
        SELECT ArtistId  
        FROM Artist  
        WHERE Name = 'U2'  
    )  
);
```

C1.21 Cuántas canciones hay de cada género (ID).

```
SELECT count(*), GenreId  
FROM Track  
GROUP BY GenreId;
```

C1.22 Cuántas canciones hay de cada género (Nombre).

```
SELECT count(*), g.Name  
FROM Track t, Genre g  
WHERE t.GenreId = g.GenreId  
GROUP BY g.Name;
```

C1.23 Canciones compradas por país.

```
SELECT count(*), i.BillingCountry  
FROM InvoiceLine il, Invoice i  
WHERE il.InvoiceId = i.InvoiceId  
GROUP BY i.BillingCountry;
```

1.3 Práctica 2

C2.1 Ciclistas con edad menor o igual a 25 años.

```
SELECT dorsal, nombre
FROM CICLISTA
WHERE edad <= 25;
```

C2.2 Puertos de categoría 'E' (Especial).

```
SELECT nompuerto, altura
FROM PUERTO
WHERE categoria = 'E';
```

C2.3 Etapas con salida y llegada en la misma ciudad.

```
SELECT netapa
FROM ETAPA
WHERE salida = llegada;
```

C2.4 ¿Cuántos ciclistas hay?

```
SELECT COUNT(*)
FROM CICLISTA;
```

C2.5 Ciclistas con edad superior a 25 años.

```
SELECT COUNT(*)
FROM CICLISTA
WHERE edad > 25;
```

C2.6 ¿Cuántos equipos hay?

```
SELECT COUNT(*)
FROM EQUIPO;
```

C2.7 Media de edad de los ciclistas.

```
SELECT AVG(EDAD)
FROM CICLISTA;
```

C2.8 Altura mínima y máxima de los puertos.

```
SELECT MIN(ALTURA), MAX(ALTURA)
FROM PUERTO;
```

C2.9 Puertos ganados por el equipo 'Banesto'.

```
SELECT p.nompuerto, p.categoria
FROM PUERTO p, CICLISTA c
WHERE p.dorsal = c.dorsal AND c.nomeq = 'Banesto';
```

C2.10 Puertos, su etapa y kilómetros.

```
SELECT p.nompuerto, p.netapa, e.km
FROM PUERTO p, ETAPA e
WHERE p.netapa = e.netapa;
```

C2.11 Equipos con algún ciclista mayor de 33 años.

```
SELECT DISTINCT e.nomeq, e.director
FROM EQUIPO e, CICLISTA c
WHERE e.nomeq = c.nomeq AND c.edad > 33;
```

C2.12 Etapas que no comienzan donde acabó la anterior.

```
SELECT e.netapa
FROM ETAPA e
WHERE e.salida <> (SELECT llegada
                  FROM ETAPA
                  WHERE netapa = e.netapa - 1);
```

C2.13 Etapas que tienen puertos de montaña.

```
SELECT DISTINCT e.netapa, e.km
FROM ETAPA e, PUERTO p
WHERE e.netapa = p.netapa;
```

C2.14 Etapas sin puertos de montaña.

```
SELECT netapa, salida
FROM ETAPA
WHERE netapa NOT IN (SELECT netapa
                   FROM PUERTO);
```

C2.15 Edad media de los ganadores de etapa.

```
SELECT AVG(edad)
FROM CICLISTA
WHERE dorsal IN (SELECT dorsal
                FROM ETAPA);
```

C2.16 Puertos con altura superior a la media.

```
SELECT nompuerto
FROM PUERTO
WHERE altura > (SELECT AVG(altura)
               FROM PUERTO);
```

C2.17 Salida y llegada de etapas con puertos de mayor pendiente.

```
SELECT salida, llegada
FROM ETAPA WHERE netapa IN (
    SELECT netapa
    FROM PUERTO
    WHERE pendiente = (SELECT MAX(pendiente)
                       FROM PUERTO)
);
```

C2.18 Ciclistas que han ganado los puertos de mayor altura.

```
SELECT dorsal, nombre
FROM CICLISTA
WHERE dorsal IN (
    SELECT dorsal
    FROM PUERTO
    WHERE altura = (SELECT MAX(altura)
                     FROM PUERTO)
);
```

C2.19 Ciclista más joven.

```
SELECT nombre
FROM CICLISTA
WHERE edad = (SELECT MIN(edad)
              FROM CICLISTA);
```

C2.20 Ciclista más joven ganador de etapa.

```
SELECT nombre
FROM CICLISTA
WHERE dorsal IN (SELECT dorsal
                 FROM ETAPA)
ORDER BY edad ASC LIMIT 1;
```

C2.21 Ciclistas que han ganado más de un puerto.

```
SELECT nombre
FROM CICLISTA
WHERE dorsal IN (
    SELECT dorsal
    FROM PUERTO
    GROUP BY dorsal
    HAVING count(*) > 1
);
```

C2.22 Etapas >190km con al menos 2 puertos.

```
SELECT e.netapa, e.salida, e.llegada
FROM ETAPA e
WHERE e.km > 190 AND (SELECT count(*)
                     FROM PUERTO p WHERE p.netapa = e.netapa) >= 2;
```

C2.23 Puertos por etapa (cantidad).

```
SELECT netapa, count(*)
FROM PUERTO
GROUP BY netapa;
```

C2.24 Número de ciclistas por equipo.

```
SELECT nomeq, count(*)
FROM CICLISTA
GROUP BY nomeq;
```

C2.25 Equipos con >3 ciclistas y edad media <29.

```
SELECT e.director, e.nomeq
FROM EQUIPO e, CICLISTA c
WHERE e.nomeq = c.nomeq
GROUP BY e.nomeq
HAVING count(*) > 3 AND AVG(c.edad) <= 29;
```

C2.26 Ganadores de etapa en equipos grandes (>5 corredores).

```
SELECT c.nombre, count(e.netapa) FROM CICLISTA c, ETAPA e
WHERE c.dorsal = e.dorsal
AND c.nomeq IN (
    SELECT nomeq FROM CICLISTA GROUP BY nomeq HAVING count(*) > 5
)
GROUP BY c.nombre;
```

C2.27 Equipo con la mayor edad media.

```
SELECT nomeq, AVG(edad)
FROM CICLISTA
GROUP BY nomeq
ORDER BY AVG(edad) DESC LIMIT 1;
```

C2.28 Equipos sin ciclistas.

```
SELECT nomeq
FROM EQUIPO
WHERE nomeq NOT IN (SELECT DISTINCT nomeq
                    FROM CICLISTA);
```

C2.29 Equipos donde todos son mayores de 29 años.

```
SELECT nomeq, director
FROM EQUIPO
WHERE nomeq NOT IN (
    SELECT nomeq
    FROM CICLISTA
    WHERE edad <= 29
);
```

C2.30 Equipos que NO han ganado etapas.

```
SELECT nomeq
FROM EQUIPO
WHERE nomeq NOT IN (
    SELECT DISTINCT c.nomeq
    FROM CICLISTA c, ETAPA e
    WHERE c.dorsal = e.dorsal
);
```

PARTE 2

Python

2.1

Práctica 1. Introducción al Entorno de Programación

» 1. Pedir nombre y edad al usuario.

Diseñar un programa en el cual se pida al usuario su nombre y su edad y muestre por pantalla la siguiente frase: <nombre> tiene <edad> años.

```
nombre = input("Ingrese su nombre: ")
edad = input("Ingrese su edad: ") #No es necesario usar int()

print(nombre, "tiene", edad, "años.")
```

» 2. Calcular el número de ladrillos.

Calcular el número de ladrillos necesarios para construir una casa, del tipo de las que se dibujan de niño. los parámetros son:

- Longitud del ancho de la fachada.
- Longitud del fondo
- Altura de las paredes
- Altura extra del techo.
- Número de puertas (todas las puertas son de 1x2 m)
- Número de ventanas (todas las ventanas son de 1x1 m)
- Dimensión del ladrillo (10cm x 5cm)

```
ancho=float(input("Ingrese la longitud del ancho de su fachada: "))
fondo=float(input("Ingrese la longitud del fondo: "))
altura_paredes = float(input("Ingrese la altura de las paredes: "))
altura_extra_techo= float(input("Ingrese la altura extra del techo: "))
puertas=int(input("Ingrese el número de puertas: "))
ventanas=int(input("Ingrese el número de ventanas: "))
#####
area_puertas = puertas * 1 * 2 #m^2
area_ventanas = ventanas * 1 * 1 #m^2
area_ladrillo = 10e-2 * 5e-2 #m^2

ladrillos_puertas = area_puertas / area_ladrillo #ud
ladrillos_ventanas = area_ventanas / area_ladrillo #ud

area_paredes = 2* (ancho * altura_paredes) + 2* (fondo * altura_paredes) #m^2
ladrillos_paredes = (area_paredes/area_ladrillo) - ladrillos_puertas -
↳ladrillos_ventanas #ud
```

```
area_techo = fondo*altura_extra_techo/2 #m^2
ladrillos_techo= area_techo/area_ladrillo #ud

total = ladrillos_paredes + ladrillos_techo #ud

print("El número total de ladrillos necesarios es:", int(total)) #Para que no muestre
↪decimales
```

2.2 Práctica 2. Tipos de Datos

» 1. Calculadora básica

Cree un programa en el que:

- Se solicite al usuario que introduzca dos números enteros por teclado.
- Se calculen la suma, la resta, el producto y la división de los dos enteros introducidos.
- Se calculen la división entera y el resto de la división entera. - Se muestren los resultados de todas las operaciones por pantalla, indicando a que operación corresponden. Pruebe a introducir números muy grandes, cercanos al máximo del tipo de datos que elija. Pruebe también a realizar una división por cero.

Nota: De forma predeterminada, la función `input()` convierte la entrada en una cadena de texto, aunque se escriba un número. Si se intenta hacer operaciones, se producirá un error.

Si se quiere que Python interprete la entrada como otro tipo de dato, se debe hacer un casting o conversión de tipo al tipo deseado.

- Si se desea leer un entero se debe hacer una conversión al tipo `int`, poniendo `int( )` al dato o expresión que se desea convertir.
Ejemplo: `a= int( input("Introduzca un número entero: ") )`.
- Si se desea leer un entero se debe hacer una conversión al tipo `float`, poniendo `float( )` al dato o expresión que se desea convertir.
Ejemplo: `a= float( input("Introduzca un número decimal: ") )`

```
texto = input("Dime un numero")
a = int(texto)
b = int(input("Dime otro número"))

suma= a+b
resta= a-b
multiplicación= a*b

división= a/b
divent= a//b
restodiv= a%b

print("La suma es", suma, ".")
print("La resta es", resta, ".")
print("La multiplicación es ", multiplicación, ".")
print("la división es" ,división, ".")
print("La división entera es" ,divent, ".")
print("El resto de la división es" ,restodiv, ".")
```

» 2. Calculadora básica 2

Cree un programa en el que:

- Se solicite al usuario que introduzca un número entero por teclado.
- Se calculen el valor absoluto, la raíz cuadrada y el logaritmo en base 10 del número introducido.
- Se muestren los resultados de todas las operaciones por pantalla, indicando a que operación corresponden.

Para poder realizar operaciones matemáticas complejas, es posible reutilizar módulos existentes que nos proporcionen las funciones o herramientas necesarias. Para el caso de operaciones matemáticas vamos a utilizar el módulo “math”. Este módulo proporciona acceso a las funciones y constantes matemáticas definidas en el estándar de C. para más información consultar la documentación en <https://docs.python.org/es/3/library/math.html>.

```
import math

n1 = int(input("Coef. a ="))
n2 = int(input("Coef. b ="))

print("1=sumar")
print("2=restar")
print("3=multiplicar")
print("4=dividir")
print("5=residuo")

op = int(input("Elige una opción:"))

if op == 1:
    suma = n1 + n2
    print("La suma es", suma)
elif op == 2:
    resta = n1 - n2
    print("La resta es", resta)
elif op == 3:
    multiplicar = n1*n2
    print("La multiplicación es", multiplicar)
elif op == 4:
    dividir = n1/n2
    print("La división es", dividir)
elif op == 5:
    residuo = n1%n2
    print("El residuo es", residuo)
```

» 3. Conversor de ángulos a radianes

Añada un programa en el que:

- Se solicite al usuario que introduzca un valor real (float) por teclado. Este valor representará un ángulo medido en grados sexagesimales.
- Transforme este ángulo a radianes: $\text{angRad} = (a * \text{PI})/180$
- Calcule el seno, coseno y tangente del ángulo transformado y muestre los resultados por pantalla.

Pruebe al menos con los valores correspondientes a 0, 30, 45 y 90 grados.

Nota: Observe que el módulo math también proporciona funciones trigonométricas y el valor de la constante PI.

```
import math
grados = float(input("Introduce el angulo en grados"))
radianes = (grados*math.pi)/180

print("Este es el seno",seno,"\\nEste es el coseno",coseno,"\\nEsta es la_
↪tangente",tangente,)
```

» 4. Ecuación.

Añada un programa en el que:

- Se soliciten por teclado tres enteros: a, b y c, que serán los coeficientes de la ecuación de segundo grado:

$$y = ax^2 + bx + c$$

- Se calculen y se muestren por pantalla los valores de x que resuelven la ecuación (solución de la ecuación).

Para una ecuación cuadrática con coeficientes reales o complejos existen siempre dos soluciones, no necesariamente distintas, llamadas raíces, que pueden ser reales o complejas (si los coeficientes son reales y existen dos soluciones no reales, entonces deben ser complejas conjugadas). Fórmula general para la obtención de raíces:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \Rightarrow \begin{cases} x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \\ x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \end{cases}$$

Pruebe con diferentes valores de los coeficientes. (ejemplo: 3,-5,6).

```
import math

print("---Ingrese las variables a,b,c de esta ecuación: y=ax^2 + bx + c---")
a= int(input("Ingrese valor a:"))
b= int(input("Ingrese valor b:"))
c= int(input("Ingrese valor c:"))

if a == 0:
    print("No es una ecuación cuadrática")
else:
    discriminante= ((-4)*a * c)
    if discriminante < 0:
        print("El discriminante es menor que 0 y no tiene soluciones reales")
    else:
        sol1=(-b + math.sqrt(discriminante)) / (2*a)
        sol2=(-b + math.sqrt(discriminante)) / (2*a)

        print("La primera solucion es x1=",sol1,"\nLa segunda solucion es x2=",sol2)
```

» 5. Texto.

Añada un programa que produzca exactamente la salida siguiente:

Todas las islas, incluso las conocidas,
son desconocidas mientras
no desembarquemos en ellas.

```
print("Todas las islas, incluso las conocidas,\nson desconocidas mientras \nnno_
↳desembarquemos en ellas.")
```

» 6. Intercambio circular de valores.

Añada un programa en el que: - Se soliciten por teclado tres valores del mismo tipo (por ejemplo, enteros): a, b y c. - Se realice el intercambio circular de valores:

B toma el valor de A,

C toma el valor de B,

A toma el valor de C.

```
print("Ingrese las variables a, b y c:")
a=input("a:")
b=input("b:")
c=input("c:")

print("Estos son los valores originales \na=",a,"\nb=",b,"\nc=",c)

a_nueva=c
b_nueva=a
c_nueva=b

print("Estos son los valores ahora: \na=",a_nueva,"\nb=",b_nueva,"\nc=",c_nueva)
```

2.3

Práctica 3. Estructuras de Control de Condicionales

» 1. Ecuación avanzada.

Añada un programa en el que:

- Partiendo de la solución del Ejercicio 4 de la práctica 2 (resolver ecuación)
- Muestre el mensaje “La ecuación tiene raíces complejas” si las raíces de la ecuación son complejas.

Nota: Si el discriminante $b^2 - 4ac$ es negativo, entonces no hay soluciones reales de la ecuación. (tiene raíces complejas)

```
import math

a=float(input("Coef. a"))
b=float(input("Coef. b"))
c=float(input("Coef. c"))

r=b**2-4*a*c

if r>=0:
    x1=(-b+math.sqrt(r))/(2*a)
    x2=(-b-math.sqrt(r))/(2*a)

    print("X1 =", x1)
    print("X2 =", x2)
else:
    print("No hay soluciones reales")
```

» 2. Calculadora avanzada.

Añada un programa en el que:

- Se solicite al usuario que introduzca dos enteros por teclado.
- Después de introducir los operandos pida mediante un código numérico la operación a realizar. Por ejemplo:
 - Pulse: (1) para sumar, (2) para restar, (3) para multiplicar, (4) para dividir, y (5) para residuo.
- Se realice la operación seleccionada y se muestre el resultado por pantalla. En las operaciones de división y residuo, si el segundo operando es igual a cero debe mostrarse el mensaje "Imposible dividir por cero".

```
n1=int(input("Primer número"))
n2=int(input("Segundo número"))

print("1 = Sumar")
print("2 = Restar")
print("3 = Multiplicar")
print("4 = Dividir")
print("5 = Residuo")

e=int(input("Elige una opción"))

if e==1:
    suma=n1+n2
    print("La suma es:",suma)
elif e==2:
    resta=n1-n2
    print("La suma es:",resta)
elif e==3:
    multiplicacion=n1*n2
    print("La multiplicación es:",multiplicacion)
elif e==4:
    if n2==0:
        print("No se puede dividir por 0")
    else:
        division=n1/n2
        print("La suma es:",division)
elif e==5:
    if n2==0:
        print("No se puede dividir por 0")
    else:
        residuo=n1%n2
        print("El residuo es:",residuo)
else:
    print("Opción inválida")
```

» 3. Desglose billetes y monedas

Realice un programa que calcule el desglose mínimo en billetes y monedas de una cantidad exacta de euros. Hay billetes de 500, 200, 100, 50, 20, 10 y 5 € y monedas de 2 y 1 €.

- 2 billetes de 200
- 1 billete de 20
- 1 billete de 10
- 2 moneda de 2

Nota: realice divisiones enteras entre las cantidades desde las más altas a las más bajas.

```
cant=int(input("Cantidad a desglosar"))

b500= cant//500
cant=cant%500

b200= cant//200
cant=cant%200

b100= cant//100
cant=cant%100

b50=cant//50
cant=cant%50

b20=cant//20
cant=cant%20

b10=cant//10
cant=cant%10

b5=cant//5
cant=cant%5

b2=cant//2
cant=cant%2

b1=cant//1

if b500 >0:
    print(b500,"Billetes de 500€")

if b200 >0:
    print(b200,"Billetes de 200€")

if b100 >0:
    print(b100,"Billetes de 100€")

if b50 >0:
    print(b50,"Billetes de 50€")

if b20 >0:
    print(b20,"Billetes de 20€")

if b10 >0:
    print(b10,"Billetes de 10€")

if b5 >0:
    print(b5,"Billetes de 5€")

if b2 >0:
    print(b2,"Monedas de 2€")

if b1 >0:
    print(b1,"Monedas de 1€")
```

» 4. Condiciones

Indica qué valores o rangos de valores provocan la aparición de los distintos mensajes:

```
dia=int(input("¿Qué día es hoy (número del mes)?")  
if 0 < dia <= 15:  
    print("Le corresponde el lado izquierdo de la calle para aparcar")  
else:  
    if dia<32:  
        print("Le corresponde el lado derecho de la calle para aparcar")  
    else:  
        print("Día incorrecto")
```

- El mensaje “Le corresponde el lado izquierdo de la calle para aparcar” aparece si el valor introducido es mayor que 0 y menor o igual que 15.
- El mensaje “Le corresponde el lado derecho de la calle para aparcar” aparece si el valor introducido es mayor que 15 y menor que 32.
- El mensaje “Día incorrecto” aparece si el valor introducido es menor o igual que 0 o mayor o igual que 32.

» 5. Día siguiente

Construye un programa que lea una fecha en forma de tres números positivos (día, mes, año) e imprima la fecha del día siguiente (año no bisiesto).

Por ejemplo:

19/6/2005 → 20/6/2005

30/4/1990 → 1/5/1990

31/12/2010 → 1/1/2011

```
d=int(input("Introduce el día: "))
m=int(input("Introduce el mes: "))
a=int(input("Introduce el año: "))

d=d+1
#Meses de 31 días
if m==1 or m==3 or m==5 or m==7 or m==8 or m==10:
    if d>31:
        d=1
        m=m+1
        print(d,m,a)
    else:
        print(d,m,a)
elif m==2:
    if d>28:
        d=1
        m=m+1
        print(d,m,a)
    else:
        print(d,m,a)
elif m==12:
    if d>31:
        d=1
        m=1
        a=a+1
        print(d,m,a)
    else:
        print(d,m,a)
else:
    if d>30:
        d=1
        m=m+1
        print(d,m,a)
    else:
        print(d,m,a)
```

» 6. Codificación

Construye un programa que leerá un carácter y escribirá en su lugar un carácter codificado.

- Si el carácter es letra o dígito, lo reemplazará por el siguiente carácter. De esta forma, el '1' se convierte en '2', 'C' en 'D', 'p' en 'q', etc.
- La 'Z' será reemplazada por la 'A'.
- La 'z' por 'a'.
- El '9' por '0'.
- Cualquier carácter distinto de letra o dígito se sustituirá por un asterisco.

Nota: La función `ord()` devuelve el código ascii de un caracter, y la función `chr()` convierte un código ascii en un caracter.

```
c = input("Dime un carácter: ")

if c >= '0' and c <= '9':      # Si es dígito
    if c == '9':
        s = '0'
    else:
        s = chr(ord(c) + 1)

elif c >= 'A' and c <= 'Z':    # Si es letra mayúscula
    if c == 'Z':
        s = 'A'
    else:
        s = chr(ord(c) + 1)

elif c >= 'a' and c <= 'z':    # Si es letra minúscula
    if c == 'z':
        s = 'a'
    else:
        s = chr(ord(c) + 1)

else:                          # Cualquier otro carácter
    s = '*'

print("Carácter codificado:", s)
```

2.4

Práctica 4. Estructuras de Control Iterativas

» 1. Calculadora avanzada 2.

Añada un programa en el que:

- Se solicite al usuario que introduzca dos enteros por teclado.
- Después de introducir los operandos pida mediante un código numérico la operación a realizar. Por ejemplo:
 - Pulse: (1) para sumar, (2) para restar, (3) para multiplicar, (4) para dividir, (5) para residuo.
 - Pulse (0) para salir
- Se realice la operación seleccionada y se muestre el resultado por pantalla. En las operaciones de división y residuo, si el segundo operando es igual a cero debe mostrarse el mensaje “Imposible dividir por cero”.
- Una vez realizada la operación seleccionada se muestre de nuevo el menú de selección de operación.
- El programa no termina hasta que se seleccione la opción de salir.

```
e = 1
while e != 0:

    print("0-salir")
    print("1-sumar")
    print("2-restar")
    print("3-multiplicar")
    print("4-dividir")
    print("5-residuo")

    e = int(input("Elige una opción"))

    if e != 0 and (e >= 1 and e <= 5):
        n1 = int(input("Primer numero: "))
        n2 = int(input("Segundo numero: "))

        if e == 1:
            suma = n1 + n2
            print("La suma es", suma)
        elif e == 2:
            resta = n1 - n2
            print("La resta es", resta)
        elif e == 3:
            mult = n1 * n2
            print("La multiplicación es", mult)
        elif e == 4:
            if n2 != 0:
                divi = n1 / n2
                print("La división es", divi)
            else:
                print("No se puede dividir por 0")
        elif e == 5:
            if n2 != 0:
                resid = n1 % n2
                print("El residuo es", resid)
            else:
                print("No se puede dividir por 0")
    else:
        print("Opción inválida")
```

» 2. Sensor de ruido

Cuando el sensor X se pone en marcha registra valores de ruido, finalizando cuando el ruido es 0. Para un estudio especial solo interesan saber la suma de aquellos valores pares registrados. Se pide:

- Simular el funcionamiento del sensor, mediante un programa que genere números enteros aleatorios en rango [0,N] hasta que el número generado sea el cero. (N se solicita por teclado.)
- El programa debe mostrar por pantalla la secuencia de números generados
- Al finalizar de generar valores, el programa debe mostrar la suma de los valores pares que se generaron.

```
import random

n = int(input("Numero máximo del rango: "))
#("para contar la de numeros aleatorios que han salido hasta llegar a 0",k)
k = 0
sumapares = 0
z = None #poner valor cualquiera que sea distinto de 0. (Para que la condición del
↪while se cumpla).
while z != 0:
    k = k + 1
    z = random.randint(0, n)
    if z % 2 == 0:
        sumapares = sumapares + z
    print (z, end=", ")

print("\nHan salido", k, "numeros")
print("La suma de todos los numeros pares es: ", sumapares)
```

8, 5, 2, 8, 3, 5, 1, 5, 0,

Han salido 9 numeros

La suma de todos los numeros pares es: 18

» 3. Número perfecto

Añada un programa en el que:

- Se pida un número N positivo por teclado (en caso de no ser positivo volver a pedir hasta que lo sea).
- Muestre por pantalla los números perfectos existentes entre 1 y N.

Nota: Un número perfecto es aquel cuya suma de sus divisores (sin incluir el mismo número) es igual al mismo número. Por ejemplo, el número 6 es un número perfecto, $6=1+2+3$.

```
N = int(input("Introduzca un número positivo: "))

# 1. Validación: Aseguramos que sea mayor que 0
while N <= 0:
    print("Número no positivo.")
    N = int(input("Introduzca un número positivo: "))

print("Buscando números perfectos entre 1 y",N,"...")

# 2. BUCLE EXTERNO: Probamos cada número (k) desde 1 hasta N
for k in range(1, N + 1):

    suma = 0 # ;IMPORTANTE! Reiniciamos la suma para cada número nuevo

    # 3. BUCLE INTERNO: Buscamos los divisores de 'k'
    # Solo probamos hasta k (sin incluirlo)
    for i in range(1, k):
        if k % i == 0:
            suma = suma + i

    # 4. Comprobamos si 'k' es perfecto
    if suma == k:
        print(k, "es un número perfecto")
```

Buscando números perfectos entre 1 y 1000 ...

6 es un número perfecto

28 es un número perfecto

496 es un número perfecto

```
#Método del profesor
for n in range(2, 1000):

    #(suma divisores, sd)

    sd = 1
    d = 2
    while d < n:
        if n % d == 0:
            sd = sd + d
        d = d + 1

    if sd == n:
        print("el numero", n, "es perfecto")
```

» 4. Número primo

Añada un programa en el que:

- Se pida un número N positivo por teclado (en caso de no ser positivo volver a pedir hasta que lo sea).
- Muestre por pantalla los números primos existentes entre 1 y N.

Nota: Un número primo divisible solo entre 1 y él mismo. Por ejemplo, el número 7 es un número primo.

```
N = int(input("Introduzca un número positivo: "))

# Validación: Aseguramos que sea mayor que 0
while N <= 0:
    print("Número no positivo.")
    N = int(input("Introduzca un número positivo: "))

print(f"Números primos entre 1 y {N}:")

# Bucle externo: Vamos probando cada número 'i' desde 2 hasta N
for i in range(2, N):

    contador = 0 # Reiniciamos el contador para cada número nuevo

    # Bucle interno: Contamos cuántos divisores tiene 'i'
    # Solo necesitamos buscar divisores hasta 'i', no hasta 'N'
    for j in range(1, i + 1):
        if i % j == 0:
            contador = contador + 1

    # --- ZONA DE COMPROBACIÓN (Fuera del bucle interno j) ---
    # Si al terminar de buscar solo tiene 2 divisores (1 y él mismo), es primo.
    if contador == 2:
        print(i, end=" ")
```

Números primos entre 1 y 7:

2 3 5

```
maxvalor = int(input("Maximo valor: "))

for n in range(2, maxvalor):
    ###Vamos a comprobar si n es primo
    ###Buscamos si hay algun divisor entre 2 y n

    d = 2
    while n % d != 0:
        d = d + 1
    if d == n:
        print("El numero", n, "es primo")
```

El numero 2 es primo

El numero 3 es primo

El numero 5 es primo

2.5

Práctica 5. Funciones I

» 1 . Calculadora avanzada 3.

Defina las siguientes funciones:

- función sumar que reciba dos números como argumento y devuelva la suma de ambos.
- función restar que reciba dos números como argumento y devuelva la resta de ambos.
- función multiplicar que reciba dos números como argumento y devuelva la multiplicación de ambos.
- función dividir que reciba dos números como argumento y devuelva la división de ambos. Si el segundo operando es igual a cero debe mostrarse el mensaje “Imposible dividir por cero” y no realizar la división.

Añada un programa en el que:

- Muestre el siguiente menú de operaciones: Pulse: (1) para sumar, (2) para restar, (3) para multiplicar, (4) para dividir, (0) para salir.
- Pida dos valores al usuario y haciendo uso de las funciones definidas, realice la operación correspondiente y muestre el resultado.
- Una vez realizada la operación seleccionada se muestre de nuevo el menú de selección de operación.
- El programa no termina hasta que se seleccioné la opción de salir.

```
def sumar(n1, n2):
    return n1 + n2

def restar(n1, n2):
    return n1 - n2

def multiplicar(a, b):
    return a * b

def dividir(dividendo, divisor):
    if divisor == 0:
        return "Error: No se puede dividir entre 0"
    else:
        return dividendo / divisor

# Iniciamos con un valor que permita entrar al bucle (diferente de 0)
opcion = -1

while opcion != 0:
    print("\n--- MENÚ ---")
    print("0 - salir")
    print("1 - sumar")
    print("2 - restar")
    print("3 - multiplicar")
    print("4 - dividir")

    opcion = int(input("Elige una opción: "))

    # 1. COMPROBAMOS SI ES UNA OPCIÓN DE CÁLCULO (1 al 4)
    # Solo entramos aquí si quiere operar. Si pone 0 o un número inválido, se salta
    → esto.
    if opcion >= 1 and opcion <= 4:
        x = int(input("Dame un numero: "))
        y = int(input("Dame otro numero: "))

        if opcion == 1:
            print("Sumar =", sumar(x, y))
        elif opcion == 2:
            print("Restar =", restar(x, y))
        elif opcion == 3:
            print("Multiplicar =", multiplicar(x, y))
        elif opcion == 4:
            print("Dividir =", dividir(x, y))

    # 2. GESTIONAMOS EL CASO DE SALIDA
    elif opcion == 0:
        print("Saliendo del programa...")
        # Al llegar aquí abajo, el bucle vuelve arriba, comprueba 'while opcion != 0'
        # y como es 0, se termina solo.

    # 3. GESTIONAMOS OPCIONES INCORRECTAS
    else:
        print("Opción no válida. Inténtalo de nuevo.")
```

```
def sumar(n1, n2):
    return n1 + n2

def restar(n1, n2):
    return n1 - n2

def multiplicar(a, b):
    return a * b

def dividir(dividendo, divisor):
    if divisor == 0:
        return "Error: No se puede dividir entre 0"
    else:
        return dividendo / divisor

# --- PASO 1: MOSTRAR MENÚ Y PEDIR OPCIÓN INICIALMENTE ---
print("\n--- MENÚ ---")
print("0 - salir")
print("1 - sumar")
print("2 - restar")
print("3 - multiplicar")
print("4 - dividir")

# Aquí es donde "añades tú el número inicialmente"
N = int(input("Introduzca una opción: "))

# --- PASO 2: BUCLE PRINCIPAL (Mientras no sea 0) ---
while N != 0:
    # CASO A: Es una opción válida para calcular (1-4)
    if N > 0 and N < 5:
        x = int(input("Dame un numero: "))
        y = int(input("Dame otro numero: "))

        if N == 1:
            print("Sumar =", sumar(x, y))
        elif N == 2:
            print("Restar =", restar(x, y))
        elif N == 3:
            print("Multiplicar =", multiplicar(x, y))
        elif N == 4:
            print("Dividir =", dividir(x, y))

        # CASO B: Es un número incorrecto (ni 0 ni 1-4)
        # Nota: No necesitamos un 'elif N == 0' porque el while lo filtra arriba
    else:
        print("Opción no válida. Inténtalo de nuevo.")

    # --- PASO 3: VOLVER A PEDIR LA OPCIÓN (CRUCIAL) ---
    # Esto se hace SIEMPRE antes de terminar la vuelta del bucle.
    print("\n--- MENÚ ---")
    print("0 - salir")
    print("1 - sumar")
    print("2 - restar")
    print("3 - multiplicar")
    print("4 - dividir")
    N = int(input("Introduzca una opción: "))

print("Saliendo del programa...")
```

» 2. Funciones de máximo y mínimo I

- Defina una función que reciba tres números y devuelva el número mayor de los tres.
- Defina una función que reciba tres números y devuelva el número menor de los tres
- Añada un programa que pida tres valores al usuario y, haciendo uso de las funciones anteriores, muestre el número menor y el número mayor de los tres.

```
#Posible en el examen.
def maximo3(n1, n2, n3):
    if n1>n2 and n1>n3:
        return n1
    elif n2>n1 and n2>n3:
        return n2
    else:
        return n3

def minimo3(n1, n2, n3):
    if n1<n2 and n1<n3:
        return n1
    elif n2<n1 and n2<n3:
        return n2
    else:
        return n3

a = int(input("Primer numero:"))
b = int(input("Segundo numero:"))
c = int(input("Tercer numero:"))

mayor = maximo3(a,b,c)

print("El mayor es", mayor)

menor = minimo3(a,b,c)

print("El menor es", menor)
```

» 3. Funciones de máximo y mínimo II

- Defina una función que reciba un número arbitrario de números (no lista) y devuelva el número mayor de los tres.
- Defina una función que reciba un número arbitrario de números (no lista) y devuelva el número menor de los tres.
- Añada un programa que pida 10 valores al usuario y, haciendo uso de las funciones anteriores, muestre el número menor y el número mayor de los tres.
- Añada un programa que pida valores al usuario hasta que pulse el 0 y, haciendo uso de las funciones anteriores, muestre el número menor y el número mayor de los tres.

NO SE HACE

» 4. Función número perfecto

Defina una función que reciba un número y devuelva 1 si el número es perfecto y 0 si el número no es perfecto

Nota: Un número perfecto es aquel cuya suma de sus divisores (sin incluir el mismo número) es igual al mismo número. Por ejemplo, el numero 6 es un número perfecto, $6=1+2+3$.

Añada un programa en el que:

- Se pida números positivos por teclado hasta que se pulse el 0
- Muestre por pantalla los números que sean perfectos (haga uso de la función anterior).

```
def esPerfecto (n):  
    #Calcular si n es numero perfecto  
  
    sd = 1#Suma de divisores  
    for i in range(2, n):  
        if n%i == 0:  
            sd = sd + i  
    return sd == n  
  
    #Si es perfecto devolver True  
    #Si no es perfecto devolver False  
  
n = int(input("dime un numero: "))  
  
if esPerfecto(n):  
    print("Es perfecto")  
else:  
    print("No es perfecto")
```

» 5. Función número primo

Defina una función que reciba un número y devuelva 1 si el número es primo y 0 si el número no es primo

Nota: Un número primo divisible solo entre 1 y él mismo. Por ejemplo, el numero 7 es un número primo.

Añada un programa en el que:

- Se pida números positivos por teclado hasta que se pulse el 0
- Muestre por pantalla los números que sean primos (haga uso de la función anterior).

NO SE HACE

» 6. Función multiplicar

Defina una función que recibe un entero entre 1 y 10 y saca en pantalla su tabla de multiplicar.

Añada un programa que pide al usuario dos números n y m y llama a la función para sacar todas las tablas desde la del n hasta la del m.

NO SE HACE

» 7. Función DNI

Defina una función que dado un número de DNI devuelva la letra correspondiente. Para calcular la letra del DNI se aplica el algoritmo 23 que consiste en:

- Obtener resto de la división entre el número completo (8 cifras) del DNI entre 23.
- Asignar una letra siguiendo la siguiente correspondencia según el resto obtenido:

$0 \rightarrow T$	$1 \rightarrow R$	$2 \rightarrow W$	$3 \rightarrow A$	$4 \rightarrow G$
$5 \rightarrow M$	$6 \rightarrow Y$	$7 \rightarrow F$	$8 \rightarrow P$	$9 \rightarrow D$
$10 \rightarrow X$	$11 \rightarrow B$	$12 \rightarrow N$	$13 \rightarrow J$	$14 \rightarrow Z$
$15 \rightarrow S$	$16 \rightarrow Q$	$17 \rightarrow V$	$18 \rightarrow H$	$19 \rightarrow L$
$20 \rightarrow C$	$21 \rightarrow K$	$22 \rightarrow E$		

Defina una función que, dado un número de DNI y una letra, retorne True si el DNI es válido y False si no lo es.

- Para que un número de DNI sea válido debe tener entre 8 dígitos y la letra debe ser la correspondiente.

Escriba un programa que pida al usuario un DNI y su letra y muestre si es válido, en caso de no ser válido volver a indicar que introduzca un DNI hasta que sea válido.

NO SE HACE

2.6

Práctica 6. Funciones II

»» 1. Factorial Recursivo

- Defina una función recursiva que reciba un número y devuelva el factorial de dicho número.
- Añada un programa que pida números positivos al usuario (en caso de no ser positivo volver a pedirlo hasta que lo sea) y, haciendo uso de la función anterior, calcule y muestre el factorial de dichos números. El programa termina cuando el usuario pulse el número 0.
- Añada un programa que pida un número N y, haciendo uso de la función anterior, calcule y muestre el factorial de cada número comprendido entre 1 y N.

NO SE HACE

»» 2. Fibonacci Recursivo

- Defina una función recursiva que reciba un número y devuelva el número de la serie de Fibonacci de dicho número.
- Añada un programa que pida números positivos al usuario (en caso de no ser positivo volver a pedirlo hasta que lo sea) y, haciendo uso de la función anterior, calcule y muestre el Fibonacci de dichos números. El programa termina cuando el usuario pulse el número 0.
- Añada un programa que pida un número N y, haciendo uso de la función anterior, calcule y muestre el Fibonacci de cada número comprendido entre 1 y N.

NO SE HACE

»» 3. Potencia recursiva

- Defina una función recursiva que reciba como argumentos dos números y devuelva como resultado la potencia del primer número elevado al segundo número.
- Añada un programa que permita comprobar el funcionamiento de la función.

NO SE HACE

»» 4. Decimal a Binario

- Defina una función recursiva que reciba como argumentos un número entero positivo (suponer que es así, no hace falta comprobar) en base 10 y muestre por pantalla su correspondiente en base 2 (binario).
- Añada un programa que permita comprobar el funcionamiento de la función.

NO SE HACE

»» 5. Dígitos

- Desarrolla una función recursiva que reciba un número entero positivo y devuelva la cantidad de dígitos que tiene. Por ejemplo: 1534 → 4, devuelve que tiene 4.
- Añada un programa que permita comprobar el funcionamiento de la función

NO SE HACE

» Ejercicio Extra 1

Contar los numeros primos menores de 1000000

```
import math
import time

def esPrimo(n):
    if n <= 1:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

# Inicia el temporizador
inicio = time.perf_counter()

contador = 0
for i in range(1,1000000+1):
    if esPrimo(i):
        contador += 1
print("Hay", contador, "números primos.")

#Deten el temporizador
fin = time.perf_counter()

#Calcula y muestra el tiempo transcurrido
print("Tiempo transcurrido:", fin-inicio,"segundos")
```

Hay 78498 números primos.

Tiempo transcurrido: 3.301279000006616 segundos

» Ejercicio Extra 2

Escribe un programa en Python que implemente una “Calculadora de Propiedades Numéricas”. El programa debe contener las siguientes funciones independientes:

1. Función `SumaDigitos(n)` (Recursiva): Debe recibir un número entero y sumar sus dígitos. Si el resultado tiene más de un dígito, debe volverse a llamar a sí misma con el nuevo valor hasta obtener un único dígito (del 0 al 9).
 - Ejemplo: 2025 -> 2+0+2+5=9 (Devuelve 9).
 - Ejemplo: 1786 -> 1+7+8+6=22 -> 2+2=4 (Devuelve 4).
2. Función `factorial_recursivo(n)`: Calcula el factorial de un número llamándose a sí misma.
3. Función `factorial_iterativo(n)`: Calcula el factorial utilizando un bucle convencional (sin recursividad).

```
def SumaDigitos(n):
    sd = 0
    while n > 0:
        r = n%10
        sd += r
        n = n//10

    if sd < 10:
        return sd
    else:
        return sumaDigitos(sd)

print("Suma digitos de 2025 =", sumaDigitos(2025))
print("Suma digitos de 1786 =", sumaDigitos(1786))
```

Suma digitos de 2025 = 9

Suma digitos de 1786 = 4

```
def factorial_recursivo(n):
    if n<0:
        print("Operación inválida")
    elif n==0:
        return 1
    else:
        return n*factorial_recursivo(n-1)

print("Factorial recursivo de 5 =", factorial_recursivo(5))
```

Factorial recursivo de 5 = 120

```
def factorial_iterativo(n):
    f=1
    for i in range(2, n+1):
        f = f * i
    return f

print("Factorial iterativo de 5 =", factorial_iterativo(5))
```

Factorial iterativo de 5 = 120

2.7 Práctica 7. Listas

» Ejemplo: Manipulación básica de matrices: eliminación de filas y columnas

Implementa un programa que gestione una matriz numérica (representada como una lista de listas). Debes crear dos procedimientos que modifiquen la matriz original:

- `eliminarFila(matriz, indice)`: Debe borrar completamente la fila indicada.
- `eliminarColumna(matriz, indice)`: Debe recorrer la matriz y eliminar el elemento correspondiente a ese índice en cada una de las filas (borrado vertical).

```
def eliminarFila(matriz, fila):  
    del matriz[fila]  
  
def eliminarColumna(matriz, columna):  
    for fila in matriz:  
        del fila[columna]  
  
matriz = [[2,-1,3], [0,5,-2], [3,1,4]]  
print("Operación de eliminar fila y columna en una matriz")  
print("Antes: ",matriz)  
eliminarFila(matriz,1)  
print("Después :", matriz)  
  
matriz = [[2,-1,3], [0,5,-2], [3,1,4]]  
print("\nOperación de eliminar columna en una matriz")  
print("Antes: ", matriz)  
eliminarColumna (matriz, 1)  
print("Después: ", matriz)
```

Operación de eliminar fila y columna en una matriz

Antes: [[2, -1, 3], [0, 5, -2], [3, 1, 4]]

Después : [[2, -1, 3], [3, 1, 4]]

Operación de eliminar columna en una matriz

Antes: [[2, -1, 3], [0, 5, -2], [3, 1, 4]]

Después: [[2, 3], [0, -2], [3, 4]]

» Ejemplo: Programa para gestionar la lista de contactos

Desarrolla una aplicación de consola en Python que permita gestionar una lista de contactos personal. El programa debe almacenar los contactos en una lista de listas, donde cada sub-lista contiene: [Nombre, Teléfono, Edad].

El sistema debe funcionar mediante un menú repetitivo con las siguientes opciones:

1. Alta de contacto: Pedir datos y guardarlos. (Nota: El teléfono debe guardarse como texto para evitar problemas con ceros a la izquierda).
2. Consulta: Buscar un contacto por su nombre y mostrar todos sus datos.
3. Baja: Buscar un contacto por nombre y eliminarlo de la lista.
4. Salir: Terminar la ejecución.

```
#Lista de todos los contactos:
contactos = []

#Función que nos muestra el menú de opciones
def mostrar_menu():
    print("\n--- MENÚ DE CONTACTOS ---")
    print("1 - Añadir contacto")
    print("2 - Consultar contacto")
    print("3 - Eliminar contacto")
    print("4 - Terminar")
    print("-----")
    opcion = int(input("Elige una opción: "))
    return opcion

#Función para añadir un contacto
def anadir_contacto():
    nombre = input("Nombre: ")
    telefono = input("Teléfono: ") #Cadena de texto
    edad = int(input("Edad: "))
    contacto = [nombre, telefono, edad]
    contactos.append(contacto)

#Función para consultar un contacto
def consultar_contacto():
    nombre_buscar = input("Ingrese el nombre del contacto a buscar: ")
    for contacto in contactos:
        if contacto[0] == nombre_buscar:
            print("Contacto encontrado:")
            print("Nombre:", contacto[0])
            print("Teléfono:", contacto[1])
            print("Edad:", contacto[2])
            return
    print("Contacto no encontrado.")

#Función para eliminar un contacto
def eliminar_contacto():
    nombre_eliminar = input("Ingrese el nombre del contacto a eliminar: ")
    for contacto in contactos:
        if contacto[0] == nombre_eliminar:
            contactos.remove(contacto)
            print("Contacto eliminado.")
            return
    print("Contacto no encontrado.")

opcion = None
while opcion != 4:
    opcion = mostrar_menu()
    if opcion == 1:
        anadir_contacto()
    elif opcion == 2:
        consultar_contacto()
    elif opcion == 3:
        eliminar_contacto()
    elif opcion == 4:
        print("Saliendo del programa de contactos.")
    else:
        print("Opción no válida. Inténtalo de nuevo.")
```